García Argueta Jaime Daniel, 312104739, jaimegarciaargueta@ciencias.unam.mx
Flores González Luis Brandon, 312218342, iluis@ciencias.unam.mx

Se importaron los módulos LProp, Semanal1 y Data.List.

- Se crea la función **modelosConj** la cual devuelve la lista de estados que satisfacen a el conjunto de fórmulas pasado como parámetro. Usa la función auxiliar modConjAux y estadosConj del módulo Semanal1.hs.
- Se crea la función **satisfenConj** la cual decide verdadero si el estado pasado como primer parámetro es modelo del conjunto de fórmulas pasado como segundo parámetro. Usa la función auxiliar interpG del módulo Semanal1.hs.
- Se crea la función **satisfConj** la cual devuelve verdadero si existe una interpretación que satisfaga a todas las fórmulas en el conjunto. Usa la función modelosConj.
- Se crea la función **insatisfenConj** dual de satisfenConj, es decir, devuelve verdadero si el estado pasado como parámetro no es modelo del conjunto de fórmulas.
- Se crea la función **insatisfConj** dual de satisfConj, es decir, devuelve verdadero si el conjunto de fórmulas no tiene modelo.
- Se crea la función **fnc** que dada una fórmula devuelve una equivalente en forma normal conjuntiva. Se usa la función auxiliar fncAux y fnn.
- Se crea la función **fncConj** la cual recibe como parámetro exclusivamente una fórmula en forma normal conjuntiva(fnc) y devuelve un conjunto(lista) de cláusulas representando a la fórmula en forma normal conjuntista. Se usa la función auxiliar clConj.
- Se crea la función **interpFNCC** que devuelve True en caso de que la fórmula en forma normal conjuntiva conjuntista(fncc) sea satisfacible en el estado pasado como parámetro. Se usa la función auxiliar interpCl.
- Se crea la función **varsFNCC** la cual devuelve el conjunto de variables presentes en la fórmula en forma fncc pasada como parámetro. Se usa la función auxiliar varsCl.
- Se crea la función **estadosFNCC** devuelve una lista con todos los posibles estados de las variables presentes en la fórmula pasada como parámetro(en fncc). Se usa la función auxiliar fnccToProp y estados del módulo Semanal1.hs
- Se crea la función **modelosFNCC** la cual devuelve una lista con los estados que satisfacen a la fórmula pasada como parámetro en forma normal conjuntiva conjuntista. Se usa la función auxiliar fnccToProp.

Además de las funciones descritas en la práctica, se tuvieron que crear funciones auxiliares nuevas. Estas son 9 en total.

- **modConjAux:** Busca en una lista de estados los que satisfacen a la fórmula pasada como parámetro y devuelve una lista con estos. Se usa interpG del Semanal1.hs
- **fnn:** Dada una fórmula devuelve una equivalente en forma normal negativa fnn(fórmula donde no figuran implicaciones ni dobles implicaciones además de que las negaciones sólo se aplican a variables proposicionales).
- **fncAux:** Realiza el trabajo de transformar una fórmula en exclusivamente forma normal negativa fnn a una equivalente en forma normal conjuntiva.

- **distribuye:** construida particularmente para el penúltimo caso de la definición de la función fncAux (el caso: Or (Or p q) (Or r s)).
- **clConj:** Recibe exclusivamente cláusulas y devuelve la cláusula en forma de conjunto(lista).
- **interpCl:** Devuelve True si la cláusula en forma de conjunto pasada como parámetro es satisfacible en el estado pasado como parámetro. Se usa interp del módulo Semanal1.hs.
- **varsCl:** Devuelve el conjunto de variables de la cláusula pasada como parámetro. Se usa vars del módulo LProp.hs. Además se usa la función union que permite regresar la unión de dos listas.
- **clToProp:** Transforma una cláusula en forma de conjunto a una en forma de proposición.
- **fnccToProp:** Transforma una fórmula en forma normal conjuntiva conjuntista en su forma de proposición.

Puntos extra.

- Se crea la función **resLit** la cual hace la resolución entre una literal y una cláusula. Requiere una proposición y una lista de ellas. Además usa la función auxiliar negLiteral.
- Se crea la función auxiliar **negLiteral** la cual analiza si una literal es la negación de otra, pero no cada caso exhaustivamente.

Finalmente **se pueden poner los mismos ejemplos que se especifican en la práctica** para probar las funciones, ya que se hizo de la manera especificada. Además mostrará los resultados esperados y por otro lado todo está documentado en el código.